(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 730 217 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
04.09.1996 Bulletin 1996/36

(51) Int. Cl.[6]: **G06F 1/32**, G06F 1/20

(21) Application number: **96102841.2**

(22) Date of filing: **26.02.1996**

(84) Designated Contracting States:
DE FR GB IT NL

(30) Priority: **28.02.1995 US 395335**

(71) Applicant: **TEXAS INSTRUMENTS INC.**
Dallas, Texas 75243 (US)

(72) Inventor: **Watts, Lavaughn F.**
Temple, Texas 76502 (US)

(74) Representative: **Holt, Michael**
Texas Instruments Limited,
Kempton Point,
68 Staines Road West
Sunbury-on-Thames, Middlesex TW16 7AX (GB)

(54) Power conservation and thermal management arrangements for computers

(57) A real-time power conservation and thermal management apparatus and method for portable computers employs a monitor (40) to determine whether a CPU may rest based upon a real-time sample of the CPU activity and temperature levels and to activate a hardware selector(500, 510, 520, 530) to carry out the monitor's determination. If the monitor determines the CPU may rest, the hardware selector reduces CPU clock time (280); if the CPU is to be active, the hardware selector returns the CPU to its previous high speed clock level (330). Switching back into full operation from its rest state occurs without a user having to request it and without any delay in the operation of the computer while waiting for the computer to return to a "ready" state. Furthermore, the monitor adjusts the performance level of the computer to manage power conservation and thermal management in response to the real-time sampling of CPU activity (10) and temperature (24). Such adjustments are accomplished within the CPU cycles and do not affect the user's perception of performance and do not affect any system application software executing on the computer.

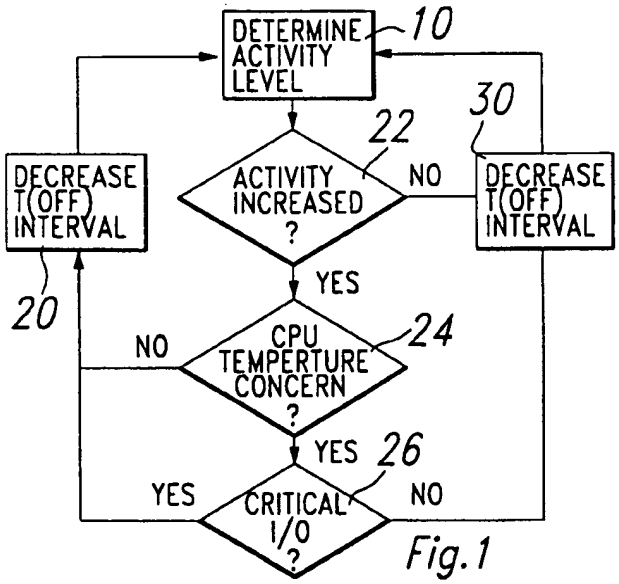

Fig.1

EP 0 730 217 A1

away from the computer for the programmed length of time. The advantage of this type of power conservation over just turning the power switch off/on is a much quicker return to full operation. However, this method of power conservation is still not real-time, intelligent power conservation while the computer is on and processing data which does not disturb the operating system, BIOS, and any third party application programs currently running on the computer.

Some attempt to meet this need was made by VLSI vendors in providing circuits that either turned off the clocks to the CPU when the user was not typing on the keyboard or woke up the computer on demand when a keystroke occurred. Either of these approaches reduce power but the computer is dead (unusable) during this period. Background operations such as updating the system clock, communications, print spooling, and other like operations cannot be performed. Some existing portable computers employ these circuits. After a programmed period of no activity, the computer turns itself off. The operator must turn the machine on again but does not have to reboot the operating system and application program. The advantage of this circuitry is like the existing "shut down" operations, a quick return to full operation without restarting the computer. Nevertheless, this method only reduces power consumption when the user walks away from the machine and does not actually extend the operational like of the battery charge.

Thermal over-heating of CPUs and other related devices is another problem yet to be addressed by portable computer manufacturers. CPUs are designed to operate within specific temperature ranges (varies depending on CPU type, manufacturer, quality, etc). CPU performance and speed degenerates when the limits of the operation temperature ranges are exceeded, especially the upper temperature range. This problem is particularly acute with CPUs manufactured using CMOS technology where temperatures above the upper temperature range result in reduced CPU performance and speed. Existing power saving techniques save power but do not measure and intelligently control CPU and/or related device temperature.

SUMMARY OF THE INVENTION

In view of the above problems associated with the related art, it is an object of the present invention to provide an apparatus and method for real-time conservation of power and thermal management for computer systems without any real-time performance degradation, such conservation of power and thermal management remaining transparent to the user.

Another object of the present invention is to provide an apparatus and method for predicting CPU activity and temperature levels and using the predictions for automatic power conservation and temperature control.

Yet another object of the present invention is to provide an apparatus and method which allows user modification of automatic activity and temperature level predictions and using the modified predictions for automatic power conservation and temperature control.

A further object of the present invention is to provide an apparatus and method for real-time reduction and restoration of clock speeds thereby returning the CPU to full processing rate from a period of inactivity which is transparent to software programs.

These objects are accomplished in a preferred embodiment of the present invention by an apparatus and method which determine whether a CPU may rest (including any PCI bus coupled to the CPU) based upon CPU activity and temperature levels and activates a hardware selector based upon that determination. If the CPU may rest, or sleep, the hardware selector applies oscillations at a sleep clock level; if the CPU is to be active, the hardware selector applies oscillations at a high speed clock level.

The present invention examines the state of CPU activity and temperature, as well as the activity of both the operator and any application software currently active. This sampling of activity and temperature is performed real-time, adjusting the performance level of the computer to manage power conservation, CPU temperature and computer power. These adjustments are accomplished within the CPU cycles and do no affect the user's perception of performance.

Thus, when the operator for the third party software of the operating system/BIOS is not using the computer, the present invention will effect a quick turn off or slow down of the CPU until needed, thereby reducing the power consumption and CPU temperature, and will promptly restore full CPU operation when needed without affecting perceived performance. This switching back into full operation from the "slow down" mode occurs without the user having to request it and without any delay in the operation of the computer while waiting for the computer to return to a "ready" state.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as other features and advantages thereof, will be best understood by reference to the detailed description with follows, read in conjunction with the accompanying drawings, wherein:

FIG. 1 is a flowchart depicting the self-tuning aspect of a preferred embodiment of the present invention.

This represents the maximum power consumption of the computer in which no power conservation measures are being used. If the CPU clock is "off" during a portion of the intervals, then there are two power levels possible for each interval. The P(on) represents the power being consumed when the clock is in its "ON" state, while P(off) represents the power being used when the clock is "OFF". If all of the time intervals in which the clock is "ON" are [is] summed into the quantity "T(on)" and the "OFF" intervals are summed into "T(off)", then it follows:

$$T = T(on) + T(off)$$

Now the energy being used during period T can be written:

$$E = [P(on) * T(on)] + [P(off) * T(off)]$$

Under these conditions, the total energy consumed may be reduced by increasing the time intervals T(off). Thus, by controlling the periods of time the clock is in its "OFF" state, the amount of energy being used may be reduced. If the T(off) period is divided into a large number of intervals during the period T, then as the width of each interval goes to zero, energy consumption is at a maximum. Conversely, as the width of the T(off) intervals increase, the energy consumed decreases.

If the "OFF" intervals are arranged to coincide with periods during which the CPU is normally inactive, then the user cannot perceive any reduction in performance and overall energy consumption is reduced from the E(max) state. In order to align the T(off) intervals with periods of CPU inactivity, the CPU activity and temperature levels are used to determine the width of the T(off) intervals in a closed loop. FIG. 1 depicts such a closed loop. The activity level of the CPU is determined at Step 10. If this level is a decrease over an immediately previous determination (Step 22), the present invention increases the T(off) interval (Step 20) and returns to determine the activity level of the CPU again. If, on the other hand, this activity level is an increase over an immediately previous determination (Step 22), a determination is made as to whether or not the temperature of the CPU is a concern (Step 24). If CPU temperature is not a concern, the present invention decreases the T(off) interval (Step 30) and proceeds to again determine the activity level of the CPU. If, on the other hand, CPU temperature is a concern, a determination is made as to whether or not the CPU is processing critical I/O, a critical function or a critical real-time event (Step 26). If critical I/O or critical function or a critical real-time event are being processed, the present invention decreases the T(off) interval (Step 30) and proceeds to again determine the activity level of the CPU. If no critical I/O is being processed, the present invention increases the T(off) interval (Step 20) and proceeds again to determine the activity level of the CPU. Thus the T(off) intervals are constantly being adjusted to match the system activity level and control the temperature level of the CPU.

Management of CPU temperature (thermal management) is necessary because CPUs are designed to operate within a specific temperature range. CPU performance and speed deteriorates when the specified high operating temperature of a CPU is exceeded (especially in CMOS process CPUs where temperatures above the high operating temperature translate into slower CPU speed). The heat output of a CPU is directly related to the power consumed by the CPU and heat it absorbs from devices and circuitry that immediately surround it. CPU power consumption increases with CPU clock speed and the number of instructions per second to be performed by the CPU. As a result, heat related problems are becoming more common as faster and increasingly complex CPUs are introduced and incorporated into electronic devices.

In any operating system, two key logic points exist: an IDLE, or "do nothing", loop within the operating system and an operating system request channel, usually available for services needed by the application software. By placing logic inline with these logic points, the type of activity request made by an application software can be evaluated, power conservation and thermal management can be activated and slice periods determined. A slice period is the number of T(on) vs. T(off) intervals over time, computed by the CPU activity and thermal levels. An assumption may be made to determine CPU activity level: Software programs that need service usually need additional services and the period of time between service requests can be used to determine the activity level of any application software running on the computer and to provide slice counts for power conservation according to the present invention. Another assumption that may be made is that each CPU has a temperature coefficient unique to that CPU - CPU temperature rise time, CPU maximum operating temperature, CPU temperature fall time and intervention time required for thermal control. If this information is not provided by the CPU manufacturer, testing of the CPU being used (or another of the same make and type tested under similar conditions) is required to obtain accurate information.

Once the CPU is interrupted during a power conservation and thermal management slice (T(off)), the CPU will save the interrupted routine's state prior to vectoring to the interrupt software. Off course, since the power conservation and thermal management software was operating during this slice, control will be returned to the active power conservation and thermal management loop (monitor 40) which simply monitors the CPU's clock to determine an exit condition for the power conservation and thermal management mode thereby exiting from T(off) to T(on) state. The interval of the next power conservation and thermal management state is adjusted by the activity level monitor, as discussed above in connection with FIG. 1. Some implementations can create an automatic exit from T(off) by the hardware logic,

the Busy_I flag (Step 210), exits the routine at RETURN I 160, and returns control to the operating system so it may continue what it was originally doing before it entered active power monitor 40.

If, however, the Power_level does not equal zero at Step 240, the routine determines whether an interrupt mask is in place. An interrupt mask is set by the system/application software, and determines whether interrupts are available to monitor 40. If interrupts are NOT_AVAILABLE, the Busy_I reentry flag is cleared and control is returned to the operating system to continue what it was doing before it entered monitor 40. Operating systems, as well as application software, can set T(on) interval to yield a continuous T(on) state by setting the interrupt mask equal to NOT_AVAILABLE.

Assuming an interrupt is AVAILABLE, monitor 40 proceeds to the SAVE POWER subroutine 250 which is fully executed during one T(off) period established by the hardware state. (For example, in the preferred embodiment of the present invention, the longest possible interval could be 18 ms, which is the longest time between two ticks or interrupts from the real-time clock.) During the SAVE POWER subroutine 250, the CPU clock is stepped down to a sleep clock level.

Once a critical I/O operation forces the T(on) intervals, the IDLE branch 60 interrupt tends to remain ready for additional critical I/O requests. As the CPU becomes busy with critical I/O, less T(off) intervals are available. Conversely, as critical I/O requests decrease, and the time intervals between them increase, more T(off) intervals are available. IDLE branch 60 is a self-tuning system based on feedback from CPU activity and temperature interrupts and tends to provide more T(off) intervals as the activity level slows and/or the CPU temperature becomes a concern. As soon as monitor 40 has completed SAVE POWER subroutine 250, shown in FIG. 2c and more fully described below, the Busy_I reentry flag is cleared (Step 210) and control is returned at RETURN I 160 to whatever operating system originally requested monitor 40.

Consider now FIG. 2c, which is a flowchart depicting the SAVE POWER subroutine 250. Monitor 40 determines what the I/O hardware high speed clock is at Step 260. It sets the CURRENT_CLOCK_RATE equal to the relevant high speed clock and saves this value to be used for CPUs with multiple level high speed clocks. Thus, if a particular CPU has 12 MHz and 6 MHz high speed clocks, monitor 40 must determine which high speed clock the CPU is at before monitor 40 reduces power so it may reestablish the CPU at the proper high speed clock when the CPU awakens. At Step 270, the Save_clock_rate is set equal to the CURRENT_CLOCK_RATE determined. Save_clock_rate 270 is not used when there is only one high speed clock for the CPU. Monitor 40 now continues to SLEEPCLOCK 280, where a pulse is sent to the hardware selector (shown in FIG. 3) to put the CPU clock to sleep (i.e., lower or stop its clock frequency). The I/O port hardware sleep clock is at much lower oscillations than the CPU clock normally employed.

At this point either of two events can happen. A system/application interrupt may occur or a real-time clock interrupt may occur. If a system/application interrupt 290 occurs, monitor 40 proceeds to interrupt routine 300, processing the interrupt as soon as possible, arming interrupt I/O at Step 310, and returning to determine whether there has been an interrupt (Step 320). Since in this case there has been an interrupt, the Save_clock_rate is used (Step 330) to determine which high speed clock to return the CPU to and SAVE POWER subroutine 250 is exited at RETURN 340. If, however, a system/application interrupt is not received, the SAVE POWER subroutine 250 will continue to wait until a real-time clock interrupt has occurred (Step 320). Once such an interrupt has occurred, SAVE POWER subroutine 250 will continue to wait until a real-time clock interrupt has occurred (Step 320). Once such an interrupt has occurred, SAVE-POWER subroutine 250 will execute interrupt loop 320 several times. If however, control is passed when the sleep clock rate was zero, in other words, there was no clock, the SAVE POWER subroutine 250 will execute interrupt loop 320 once before returning the CPU clock to the Save_clock_rate 330 and exiting (Step(340)).

Consider now FIG. 2d which is a flowchart showing ACTIVITY branch 70 triggered by an application/system activity request via an operating system service request interrupt. ACTIVITY branch 70 begins with reentry protection. Monitor 40 determines at Step 350 whether Busy_I has been set to BUSY_FLAG. If it has, this means the system is already in ACTIVITY branch 70 and cannot be interrupted. If Busy_I = BUSY_FLAG, monitor 40 exits to RETURN I 160, which is an indirect vector to an old activity vector interrupt for normal processing, via an interrupt vector after the operating system performs the requested service.

If however, the Busy_I flag does not equal BUSY_FLAG, which means ACTIVITY branch 70 is not being accessed, monitor 40 determines at Step 360 if the BUSY_A flag has been set equal to BUSY_FLAG. If so, control will be returned to the system at this point because ACTIVITY branch 70 is already being used and cannot be interrupted. If the Busy_A flag has not been set, in other words, Busy_A does not equal BUSY_FLAG, monitor 40 sets Busy_A equal to BUSY_FLAG at Step 370 so as not to be interrupted during execution of ACTIVITY branch 70. At Step 380 the Power_level is determined. If Power_level equals zero, monitor 40 exits ACTIVITY branch 70 after clearing the Busy_A reentry flag (Step 390). If however, the Power_level does not equal zero, the CURRENT_CLOCK_RATE of the I/O hardware is next determined. As was true with Step 270 of FIG. 2C, Step 400 of FIG. 2d uses the CURRENT_CLOCK_RATE if there are multiple level high speed clocks for a given CPU. Otherwise, CURRENT_CLOCK_RATE always equals the CPU high speed clock. After the CURRENT_CLOCK_RATE is determined (step 400), at Step 410 Idle_tick is set equal to the constant START_TICKS established for the previously determined CURRENT_CLOCK_RATE. T(off) intervals are established based on the current high speed clock that is active.

before the target temperature is reached. Once CPU temperature starts to lower, it is O.K. to go back to the regular thermal constant number because 1) you have selected the right slice period, or 2) the active power portion of the active power and thermal management has taken over, so the sampling rate can be reduced.

Examples of source code that can be stored in the CPU ROM or in an external RAM device, according to one embodiment of the invention, are listed in the COMPUTER PROGRAMS LISTING section under: 1) Interrupt 8 Timer interrupt service - listed on pages 31 to 36; 2) CPU Sleep Routine - listed on page 37 3) FILE=FORCE5.ASM - listed on pages 38 to 42; and 4) FILE=Thermal.EQU - listed on page 43.

Utilizing the above listed source code, and assuming that Interrupt 8 Timer interrupt service is the interrupt mask called at Step 240 of IDLE loop 60 or at Step 460 of ACTIVITY loop 70, the procedure for thermal management is set up "Do Thermal Management if needed" after which the system must decide if there is time for thermal management "Time for Thermal Management?". If there is time for thermal management, the system calls the file "force_sleep" if there is time to sleep (which also sleeps any PCI bus coupled to the CPU), or alternatively, could do a STI nop and a halt - which is an alternate way and does not get PCI devices and does not have a feedback loop from the power and temperature management systems. The "force_sleep" file gets feedback from other power systems. Force_sleep does a jump to force5.asm, which is the PCI multiple sleep program. Are there speakers busy in the system? Is there something else in the system going on from a power management point of view? Are DMAs running in the system? Sleeping may not be desirable during a sound cycle. It needs to know what is going on in the system to do an intelligent sleep. The thermal management cares about the CPU and cares about all the other devices out there because collectively they all generate heat.

There are some equations in the program that are running - others that may or may not be running. "tk" is the number of interrupts per second that are sampled times the interval that is sampled over. "it" represents a thermal read constant and the thermal read constant in the present embodiment is 5. In the code, the thermal read constant is dynamically adjusted later depending on what the temperature is. Thus, this is the starting thermal read interval, but as the temperature rises, reading should be more often and the cooler it is, reading should be less often than 5 minutes - e.g., 10 minutes. The thermal read constant will adjust. TP1 or TP2 represents what percentage of the CPU cycles do we want to sample at - for example, TP7 set at 50 = the number of interrupts that have to occur over some period of time such that if we take that number that going to represent every so many clock cycles that go by before we sample and sleep the CPU. These equations are variable. Other equations can also be used.

Thus, one concept of the present invention is that there are various levels of temperature that require testing in relationship to the hottest point to be managed. The sample period will change based on temperature and active feedback. Active feedback may be required even though thermal management has determined that the CPU temperature is too high and should be reduced (by slowing or stopping the CPU clock). CPU clock speed may not be reduced because other system things are happening - the result is intelligent feedback. The power conservation and thermal management systems asks the CPU questions such as are you doing something now that I cannot go do? If not, please sleep. If yes, don't sleep and come back to me so that I can reset my count. The result is a graduated effect up and graduated effect down and the thermal read constant time period adjusts itself in response to CPU temperature. Performance taken away from the user during power conservation and thermal management control is balanced against critical I/O going on in the system.

Active power and thermal management cooperates with standard CPU power management so that when standard power management gets a chance to take over the active feedback can start degrading even though the temperature has not. Existing power/thermal management systems turn on and stay on until the temperature goes down. Unfortunately, this preempts things in the system. Such is not the case in the environment of the present invention. The same sleep manager works in conjunction with power conservation and thermal management - the sleep manager has global control. As a example, while CPU temperature may be rising or have risen to a level of concern, the system may be processing critical I/O, such as a wave file being played. With critical I/O, the system of the present invention will play the wave file without interruption even though the result may be a higher CPU temperature. CPUs do not typically overheat all at once. There is a temperature rise gradient. The system of the present invention takes advantage of the temperature rise gradient to give a user things that affect the user time slices and take it away from him when its not affected.

Thermal management can be also be achieved using a prediction mode. Prediction mode utilizes no sensors or thermistors or even knowledge as to actual CPU temperature. Prediction mode uses a guess - i.e. that the system will need the ad hoc interrupt once every 5 seconds or 50 times/second (=constant) and then can take it up or down based on what the system is doing with the active power and thermal management. The prediction theory can also be combined with actual CPU temperature monitoring.

Once the power conservation and thermal management monitor is activated, a prompt return to full speed CPU clock operation within the interval is achieved so as to not degrade the performance of the computer. To achieve this prompt return to full speed CPU clock operation, the preferred embodiment of the present invention employs some associated hardware.

CPU clock, and is the equivalent of CPU CLOCK of FIG. 2. (If the device includes a PCI bus, the output of AND gate 770 may also be coupled to the PCI bus if it is to utilize the clock signal.)

Consider now FIG. 5, which depicts a schematic of another actual sleep hardware implementation for a system such as the Intel 80286 (CPU can have its clock stopped). The Western Digital FE3600 VLSI is used for the speed switching with a special external PAL 780 to control the interrupt gating which wakes up the CPU on any interrupt. The software power conservation according to the present invention monitors the interrupt acceptance, activating the next P(i)deltaTi interval after the interrupt.

Any interrupt request to the CPU will return the system to normal operation. An interrupt request ("INTRQ") to the CPU will cause the PAL to issue a Wake Up signal on the RESCPU line to the FE3001 (not shown) which in turn enables the CPU and the DMA clocks to bring the system back to its normal state. This is the equivalent of the "INterrupt_" of FIG. 2. Interrupt Request is synchronized to avoid confusing the state machine so that Interrupt (INT-DET) will only be detected while the cycle is active. The rising edge of RESCPU will wake up the FE 3001 which in turn releases the whole system from the Sleep Mode.

Implementation for the 386SX is different only in the external hardware and software power conservation loop. The software loop will set external hardware to switch to the high speed clock on interrupt prior to vectoring the interrupt. Once return is made to the power conservation software, the high speed clock cycle will be detected and the hardware will be reset for full clock operation.

Implementation for OS/2 uses the "do nothing" loop programmed as a THREAD running in background operation with low priority. Once the THREAD is activated, the CPU sleep, or low speed clock, operation will be activated until an interrupt occurs thereby placing the CPU back to the original clock rate.

Although interrupts have been employed to wake up the CPU in the preferred embodiment of the present invention, it should be realized that any periodic activity within the system, or applied to the system, could also be used for the same function.

While several implementations of the preferred embodiment of the invention has been shown and described, various modifications and alternate embodiments will occur to those skilled in the art. Accordingly, it is intended that the invention be limited only in terms of the appended claims.

COMPUTER PROGRAMS LISTING

1) Interrupt 8 Timer interrupt service - pages 31 to 36. Interrupt 8 Timer interrupt service is loaded onto the CPU ROM or an external RAM and is an interrupt mask that may be called at Step 240 of IDLE loop 60 or at Step 460 of ACTIVITY loop 70.

2) CPU Sleep Routine - page 37. CPU Sleep Routine is loaded onto the CPU ROM or an external RAM and is a file that may be called at Step 250 of IDLE loop 60 or ACTIVITY loop 70.

3) FILE=FORCE5.ASM - pages 38 to 42. FILE=FORCES.ASM is a PCI multiple sleep program that is loaded onto the CPU ROM or an external RAM and is a file that may be called at Step 250 of IDLE loop 60 or ACTIVITY loop 70.

4) FILE=Thermal.EQU - listed on page 43. FILE=Thermal.EQU is loaded onto the CPU ROM or an external RAM and is a file that may be called at STEP 240 of IDLE loop 60 or at Step 460 of ACTIVITY loop 70.

```
        APM_STATE_CMOS                          ; Byte to hold APM Write Flag
        out         CMOS_AD,al                  ; Output it to CMOS
        in          al,CMOS_DT                  ; and store it
;
;       Check Command Register
;
        cmp         al,80h
        jne         CheckAPMCommand1
        mov         byte ptr APMCommandCurrent,al ; Debug locations
;[6.02b]mov        power_level,0               ; Take it way - pure zero
        mov         al,8fh                      ; Completed command
WriteAPMCommand:
        out         CMOS_DT,al                  ; New command
        jmp         short APMCommandComplete
EnablePowerManagement:
        mov         byte ptr APMCommandCurrent,al ; Debug locations
        mov         al,00h                      ; command completed
        jmp         short WriteAPMCommand

CheckAPMCommand1:
        cmp         al,81h
        je          EnablePowerManagement
        cmp         al,88h
        je          APMCommandComplete          ; Waiting on Clear

        cmp         al,8fh
        je          APMCommandComplete          ; Skip Power Saving APM
        mov         ah,al
        xor         al,al
        out         CMOS_DT,al                  ; Clear it
        mov         al,ah                       ; bump count
        xor         ah,ah
        add         apm_tick_count,ax           ; done

APMCommandComplete:
;
;       Compute Interval
;
ComputeInterval:
        cmp         WORD PTR [DC_Second],0
        dec         WORD PTR [DC_Second].0
        jne         ComputeMinuteInterval
        mov         WORD PTR [DC_Second],SECOND_RELOAD

ComputeMinuteInterval:
        dec WORD PTR [DC_Minute]    ; one more tick passed, one
                                    ; tick close to full minute
        cmp         WORD PTR [DC_Minute],0      ; reached minute yet??
        je          NotTimerExit                ; yep, then update
        jmp         timer_exit                  ; nope, keep waiting
NotTimerExit:
;
;       Do Thermal Management if needed
;
        dec         ThermalMinute
        cmp         ThermalMinute,0
        jne         SkipThermalThisPass
        mov         ThermalMinute,1             ; Error Condition on Read
        cmp         LilyKBBusy,0
```

```
T0ThermalSlice:
        mov         ThermalSlice,TSLICE0

ResetThermalSlice:
        mov         TimeThermalSlie,1           ; Will execute on this slice
;
;       Fall        Thru for the rest of the story
;
OldNotTimerExit:
;
;       Setup for new number of ticks
;
        mov         WORD PTR [DC_Minute],MINUTE_RELOAD
;
;       Need to test for Thermal Reading needed
;

;
;       We must now update any change in Operational Status
;       Set up Base DS to BIOS RAM AREA
;
        mov         ax,DS40H
        mov         es,ax                       ; [5.10.c7]

;
;One minute passed, so update current system parameters:Do the Power On Times
;
        CLI
        inc         SystemRunTime               ; bump up the number of min run
;
;       Read AC Port Operations
;
        BATTERY_TEST
        jne         RunningOnAc

        inc         SystemTime                  ; Time on Battery  [5.10.c3]

        jmp         RuningCurrentSystemBattery

RunningOnAc:
;
;       Calculate last usage on AC power
;
        mov         cx,SystemRunTime            ; Total run time this session
        mov         OldState,ch                 ; [5.10.1]
        jmp         CurrentACAll
CurrentAcAll:
;
;       We are currently on AC; Was the Last Interrupt on AC?
;
        mov         cx,SystemRunTime            ; ch =Flags for Current Session
        and         ch,SESSION_STATUS
        cmp         ch,SESSION_STATUS           ; if equal last on battery
        jne         StillOnAC                   ; Still on AC, we are okay.

        APM_EVENT POWER_STATUS_CHANGE           ; On Bat/Tell APM
;
;       We must now recalculate our parameter: Session Change
;
```

```
        Setup return for our slice

        popf                                            ; Status

        pushf
        push    cs                                      ; My cs
        push    offset  ThermalSuspend                  ; My exit
        jmp     short   BATransfer
BAExitNow:
        popf
BATransfer:
        jmp     cs:dword ptr ipc_timer      ; do other chained timer routines

ThermalSlice            db      TSLICE0
TimeThermalSlice        db      0
BATempdebug             db      0AAh
ThermalSuspend:
        pushf
        push    ds
        push    cs
        pop     ds

        pusha

        mov     cx,1
BAOutsideHeatLoop:

        call    force_sleep     gets feedback from other power systems
;;        sti                           }alternate way - does not get
;;        nop                           }PCI devices does not have
;;        hlt                           }feedback loop from the
        loop    BAOutsideHeatLoop               power management systems.

        mov     al,ThermalSlice
        mov     timeThermalSlice,al

        popa


        pop     ds
        popf
        iret

timer_interrupt         endp
```

```
;
;FILE=FORCE5.ASM (LILYP ONLY)
;
busy_force     db        0
force_sleep5   proc      near
;
        test    byte ptr cs:busy_force,BUSY_FLAG
        jnz     Busy5


;       Here we are taking our turn of the cpu on this clock cycle
;
CheckBellAction5:

        cli

        APM_STATE_CMOS
        out     CMOS_AD,al
        in      al,CMOS_DT
        and     al,80h                          ; command bit on?
        cmp     al,80h
        je      BellInUse5                        ; yes, speaker busy
        in      al,PORT_61                      ; Save Port 61
        jmp     $+2                             ; Need 5 ns delay (290 ns overkill)
        and     al,LOW_BITS_61                  ;; Mask off low order bits
        cmp     al,0
        je      bell_is_off5                    ; Bell free, sleep
BellInUse5:
;[6.02b]
        and     byte ptr cs:busy_force,NOT_BUSY_FLAG
;
;       bell in use, exit
;
        sti
Busy5:


        ret
bell_is_off5:
;
;       Can we do it because there maybe DMA running
;
        in      al,08h
        mov     ah,al
        in      al,0d0h
        or      ah,al
        cmp     al,0
        jne     BellInUse5                      ; DMA Active

     or     byte ptr cs:busy_force,BUSY_FLAG
     cli
     push       cx                  ; Save loop counter
     mov        cl,02h              ; PCI Bus clock divider to set
     call       PCICONFIG           ; Set it; cx = old value to reset

     move       al,2ah
     out        0f2h,al
     in         al,0f3h             ; Get value
     push       ax                  ; Save the mother load
     and        al,01111111b
```

```
;
;FILE=FORCE5.ASM (ULYP ONLY)
;
busy_force    db    0
force_sleep5  proc   near
;
    test   byte ptr cs:busy_force,BUSY_FLAG
    jnz    Busy5

;    Here we are taking our turn of the cpu on this clock cycle
;
CheckBellAction5:

    cli

    APM_STATE_CMOS
    out    CMOS_AD,al
    in     al,CMOS_DT
    and    al,80h              ; command bit on?
    cmp    al,80h
    je     BellInUse5          ; yes, speaker busy
    in     al,PORT_61          ; Save Port 61
    jmp    $+2                 ; Need 5 ns delay (290 ns overkill)
    and    al,LOW_BITS_61      ;; Mask off low order bits
    cmp    al,0
    je     bell_is_off5        ; Bell free, sleep
BellInUse5:
;[6.02b]
    and    byte ptr cs:busy_force,NOT_BUSY_FLAG
;
;    bell in use, exit
;
    sti
Busy5:


    ret
bell_is_off5:


    or     byte ptr cs:busy_force,BUSY_FLAG
    cli
    push   cx          ; Save loop counter
    mov    cl,02h      ; PCI Bus clock divider to set
    call   PCICONFIG   ; Set it; cx = old value to reset

    mov    al,2ah
    out    0f2h,al
    in     al,0f3h     ; Get value
    push   ax          ; Save the mother load
    and    al,01111111b
    or     al,00000100B
```

```
;
;FILE=pciconf.asm
;
;-----------------------------------------------
;   Initialize PCI for Gary
;-----------------------------------------------
;
;   CX = Value to write
;   CX = Value read
;

PCI_CONFIG_ADDRESS EQU   0CF8H
PCI_CONFIG_DATA   EQU   0CFCH
PCI_CONFIG_DATA2  EQU   0CFEH


pciconfig    proc   near
.386C
     push   eax
     push   ebx
     push   dx

     mov    ax,8000h          ; BASE Addressing mode
;
;    Put the Register for PCI access in BX
;
     mov    bx,44h            ; Done - PCI Bus clock register

;
;    Access the PCI Register Set
;
     push   eax
     shl    eax,10h
     mov    ax,bx
     mov    dx,PCI_CONFIG_ADDRESS
     out    dx,eax            ; Register wanted to be selected
     mov    dx,PCI_CONFIG_DATA
     in     eax,dx            ; Read the register set wanted
     shr    eax,10h
     mov    dx,ax
     pop    eax
     push   dx
     mov    dx,PCI_CONFIG_DATA2    ;
     mov    al,cl
     pop    cx

     out    dx,al             ; Data out to PCI Wanted

     pop    dx
     pop    ebx
     pop    eax
.286C
     ret
pciconfig    endp
```

5. The arrangement of any preceding claim, wherein said central processing unit (CPU) is part of a computer.

6. An arrangement as claimed in any preceding claim and wherein said sleep manager controls periods of time said clock is in an OFF state, the length of said periods of time said clock is in an OFF state being appropriate to allow said central processing unit to operate at an optimized utilization percentage.

7. The device of Claim 16, wherein energy consumption in said device is at a maximum when the length of each period of time said clock is in an OFF state is at zero.

8. The device of Claim 16, wherein energy consumption in said device decreases as the length of each period of time said clock is in an OFF state increases.

9. A device according to any one of Claims 16, 17 or 18, wherein said periods of time said clock is in an OFF state are constantly being adjusted to optimize said utilization percentage and control the temperature of said central processing unit.

10. An arrangement according to any of Claims 16 to 19, wherein said OFF state represents the minimum clock rate at which said central processing unit can operate.

11. An arrangement according to any of Claims 16 to 20, wherein said minimum clock rate may be zero for central processing units that can have their clocks stopped.

12. An arrangement as claimed in any preceding claim wherein said CPU sleep manager further sleeps a PCI bus coupled to the device.

13. The arrangement of Claim 12 wherein said CPU sleep manager further sleeps any other CPUs connected to the PCI bus.

14. An arrangement, comprising:
a central processing unit (CPU); and
means for determining whether said central processing unit (CPU) may rest based upon the central processing unit (CPU) activity and temperature levels and activating a hardware selector(500, 510, 520, 530) based upon said determination.

15. The arrangement of Claim 14, wherein the hardware selector applies oscillations to the clock input of said central processing unit (CPU) at a slower sleep clock level if the central processing unit is to sleep or rest or at a higher full processing rate speed clock level if the central processing unit is to be active.

16. An arrangement according to any of Claims 14 or 15, wherein the hardware selector prevents the oscillations from reaching the clock input of said central processing unit (CPU) if the central processing unit is to sleep or rest or supplies oscillations at the full processing rate speed clock level if the central processing unit is to be active.

17. An arrangement as claimed in any preceding claim and wherein predictul values of activity and temperature are needed.

18. A device, comprising:
a computer;
means for predicting the activity and temperature levels within said computer; and
means for using said prediction for automatic power conservation and temperature control, said power conservation and temperature control remaining transparent to a user of said computer.

19. The arrangement of Claim 18, including means for user modification of said automatic activity and temperature level predictions and using said modified predictions for automatic power conservation and temperature control.

20. An arrangement, comprising:
a computer including a central processing unit (CPU); and
means for sampling a utilization percentage and temperature of said central processing unit (CPU); and
means for adjusting processing speed of said central processing unit (CPU) to optimize said utilization percentage.

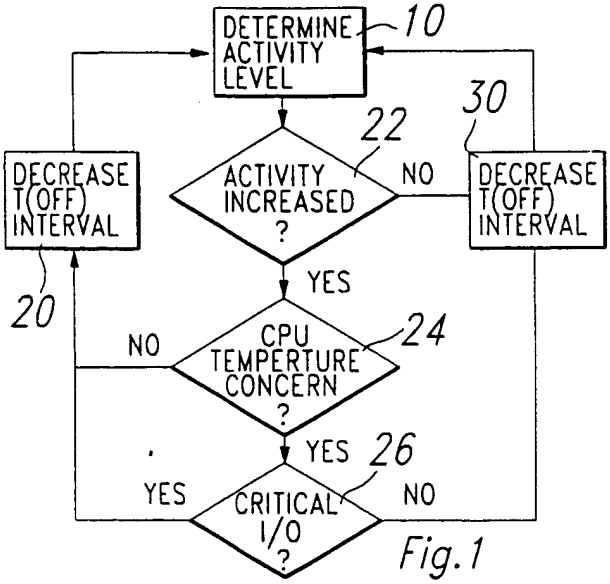

Fig.1

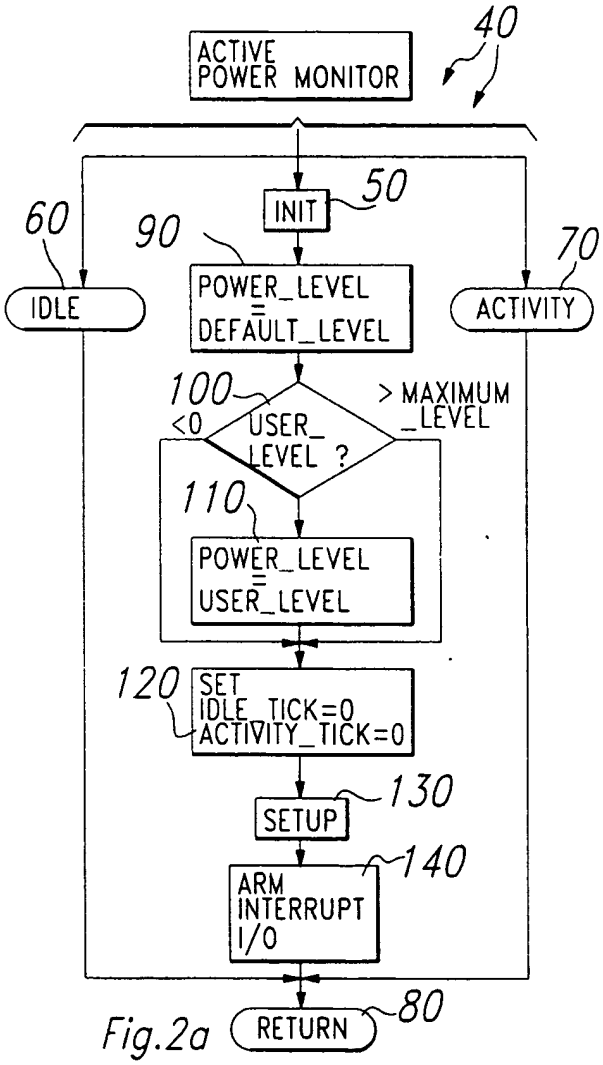

Fig.2a

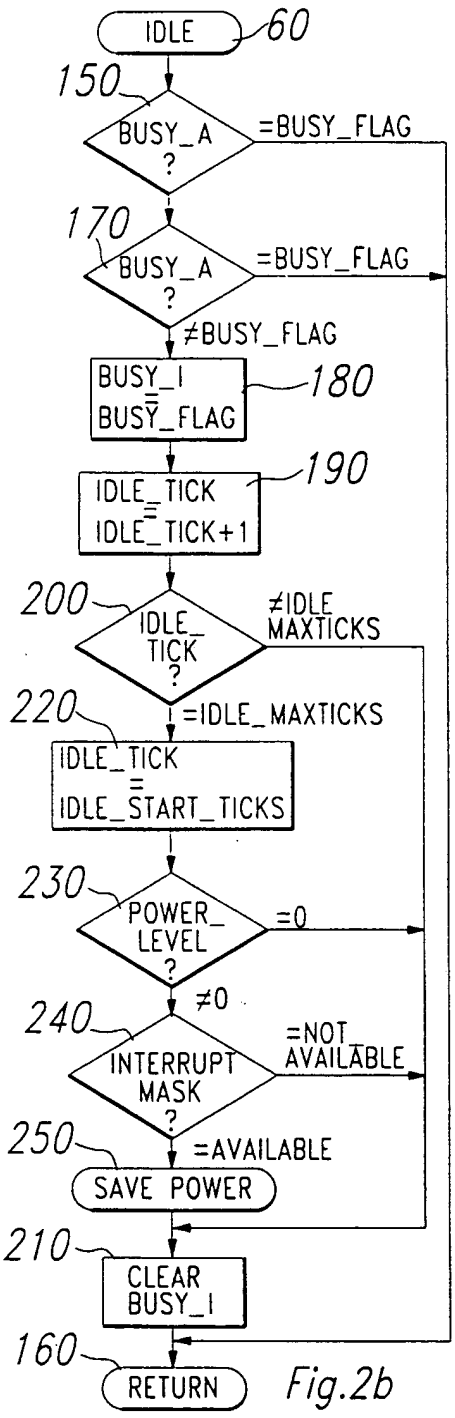

Fig.2b

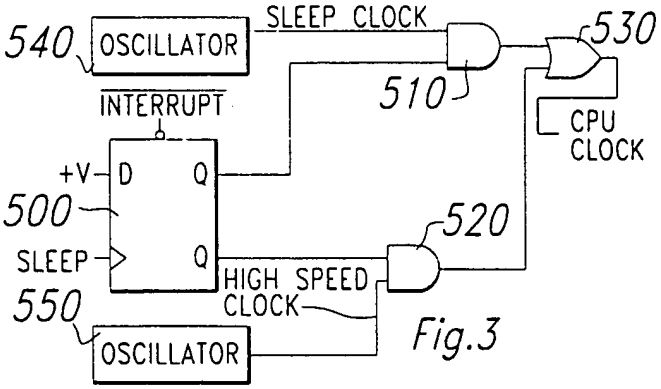

Fig.3

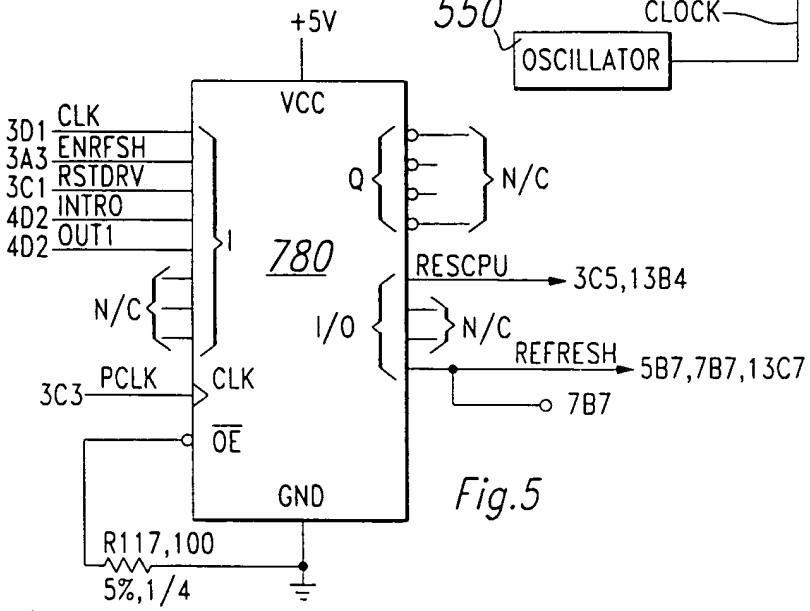

Fig.5

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 96 10 2841

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
X	EP-A-0 566 395 (DIA SEMICON SYSTEMS INC) 20 October 1993 * abstract; claims 1,2; figure 1 *	1-5,14, 15, 20-22, 24,25	G06F1/32 G06F1/20
X A	US-A-5 189 314 (GEORGIOU CHRISTOS J ET AL) 23 February 1993 * abstract; figures * * column 3, line 8 - line 38 *	14,15, 17,18 1-5, 20-22, 24,25	
Y	EP-A-0 426 410 (TEXAS INSTRUMENTS INC) 8 May 1991 * the whole document *	1-11, 14-25	
Y	US-A-5 287 292 (KENNY JOHN D ET AL) 15 February 1994 * abstract * * column 1, line 51 - column 2, line 2 * * column 3, line 47 - column 4, line 15 *	1-11, 14-25	TECHNICAL FIELDS SEARCHED (Int.Cl.6) G06F
A	EP-A-0 501 655 (IBM) 2 September 1992 * abstract; claims; figures *	1-5,11, 14-16, 20-25	
A	WO-A-92 10032 (ADAPTIVE SOLUTIONS INC) 11 June 1992 * abstract; claims 1-4,25; figures 1,2,6 *	1,2,4, 14-18,20	

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
BERLIN	14 May 1996	Durand, J

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)